

```

1 from microbit import *
2 import radio
3
4 # 無線通信の初期設定
5 radio.config(group=1, power=7)
6 radio.on()
7
8 data = [0] * 19 # 0 で初期化された長さ 19 の配列
9 # ボタンAが押されたときに配列を編集し送信する
10 while True:
11     if button_a.was_pressed():
12         mapped = scale(display.read_light_level(), from_=(0,255), to=(0,9))
13         if mapped >= 6:
14             data[1] = 1
15         else:
16             data[1] = 2
17         radio.send_bytes(bytearray(data))
18         print(bytearray(data))
19         for i in range(len(data)):
20             display.scroll(str(data[i]))

```

1台目のマイクロビットのコード

```

from microbit import *
import radio

# 無線通信の初期設定
radio.config(group=1, power=7)
radio.on()

while True:
    # 受信したメッセージを取得
    data = radio.receive_bytes()
    if data:
        data = bytearray(data)
        if data[n-1] != 0 and data[n+1] == 0:
            level = display.read_light_level() # 0~255の明るさを取得
            mapped = scale(level, from_=(0,255), to=(0,9)) # 明るさを0~9にマッピング
            # 明るさに応じてデータを変更
            if mapped >= 6:
                data[n] = 1
            else:
                data[n] = 2
            radio.send_bytes(data) # データを送信
            for i in range(len(data)):
                display.scroll(str(data[i]))
            print(data)

```

2台目以降(n台目)のマイクロビットのコード

```
1 |<!DOCTYPE html>
2 <html lang="ja">
3 <head>
4 <meta charset="UTF-8">
5 <meta charset="UTF-8">
6 <title>send (bytearray対応)</title>
7 <script src="https://www.gstatic.com/firebasejs/8.10.1/firebase-app.js"></script>
8 <script src="https://www.gstatic.com/firebasejs/8.10.1/firebase-database.js"></script>
9 </head>
10 <body>
11 <h1>send (bytearray対応)</h1>
12 <pre id="output"></pre>
13 <button id="connectBtn">micro:bit に接続</button>
14
15
16 <script>
17 // ==== Firebase 初期化 ====
18 const firebaseConfig = {
19   apiKey: [REDACTED]
20   authDomain: [REDACTED]
21   databaseURL: [REDACTED]
22   projectId: [REDACTED]
23   storageBucket: [REDACTED]
24   messagingSenderId: [REDACTED]
25   appId: [REDACTED]
26 };
27 firebase.initializeApp(firebaseConfig);
28 const database = firebase.database();
29
30
31 // ==== Web Serial 接続 ====
32 let port;
33 let reader;
34
35
36 async function connectMicrobit() {
37   try {
38     try {
39       port = await navigator.serial.requestPort();
40       await port.open({ baudRate: 115200 });
41     } catch (err) {
42       console.error("シリアルポート選択キャンセル:", err);
43       alert("micro:bit を選択してください");
44       return;
45     }
46
47     reader = port.readable.getReader();
48
```

```
49 let buffer = '';
50 const textDecoder = new TextDecoder();
51
52
53 while (true) {
54     const { value, done } = await reader.read();
55     if (done) break;
56     if (!value) continue;
57
58
59     buffer += textDecoder.decode(value, { stream: true });
60     const lines = buffer.split('\n');
61     buffer = lines.pop();
62
63
64     lines.forEach(line => {
65         line = line.trim();
66         if (!line) return;
67
68
69         console.log("受信した行:", line);
70         document.getElementById('output').textContent = "raw: " + line + "\n";
71
72
73         try {
74             let arr = [];
75
76
77             // bytearray形式の場合
78             if (line.startsWith("bytearray")) {
79                 const match = line.match(/bytearray\b'(.*)'\b/);
80                 if (match) {
81                     const hexStr = match[1];
82                     hexStr.replace(/\x([0-9A-Fa-f]{2})/g, (_, p1) => {
83                         arr.push(parseInt(p1, 16));
84                     });
85                 }
86             } else if (line.startsWith("[") ) {
87                 // JSON文字列の場合
88                 arr = JSON.parse(line);
89             } else {
```

```

90     // カンマ区切り文字列
91     arr = line.split(',').map(Number);
92 }
93
94
95     // room1~room18 に変換
96     const obj = {};
97     for (let i = 1; i < arr.length && i <= 18; i++) {
98         | obj["room" + (i)] = arr[i];
99     }
100
101
102     // タイムスタンプ追加
103     obj.timestamp = Date.now();
104
105
106     // Firebase に保存
107     database.ref('rooms/latest').set(obj);
108
109
110     // ブラウザに表示
111     document.getElementById('output').textContent += JSON.stringify(obj) + '\n';
112
113
114     } catch (e) {
115         console.error('配列→JSON変換エラー:', line, e);
116     }
117     });
118 }
119
120
121     } catch (err) {
122         console.error('接続エラー:', err);
123     }
124 }
125
126
127 document.getElementById("connectBtn").addEventListener("click", connectMicrobit);
128 </script>
129 </body>
130 </html>

```

webサイトの HTML1

```

1 <!DOCTYPE html>
2 <html lang="ja">
3 <head>
4 <meta charset="UTF-8">
5 <title>部屋の明暗表示</title>
6 <style>
7   .room {
8     display: inline-block;
9     width: 80px;
10    height: 80px;
11    margin: 5px;
12    text-align: center;
13    font-family: sans-serif;
14    border-radius: 5px;
15    color: ■white;
16    vertical-align: top;
17    padding-top: 10px;
18  }
19  .bright { background-color: ■green; }
20  .dark { background-color: □black; }
21  .other { background-color: ■gray; }
22  .room-number {
23    font-weight: bold;
24    margin-bottom: 5px;
25  }
26  .room-status {
27    font-weight: bold;
28  }
29  #timestamp {
30    margin-top: 10px;
31    font-size: 14px;
32    font-family: sans-serif;
33  }
34 </style>
35 <script src="https://www.gstatic.com/firebasejs/8.10.1/firebase-app.js"></script>
36 <script src="https://www.gstatic.com/firebasejs/8.10.1/firebase-database.js"></script>
37 </head>
38 <body>
39 <h1>部屋の明暗表示</h1>
40 <div id="roomsContainer"></div>
41 <div id="timestamp"></div>
42
43 <script>
44 // ==== Firebase 初期化 ====
45 const firebaseConfig = {
46   apiKey: ██████████
47   authDomain: ██████████

```



```
48 databaseURL: [REDACTED]
49 projectId: [REDACTED]
50 storageBucket: [REDACTED]
51 messagingSenderId: [REDACTED]
52 appId: [REDACTED]
53 };
54 firebase.initializeApp(firebaseConfig);
55 const database = firebase.database();
56
57 const container = document.getElementById('roomsContainer');
58 const tsDiv = document.getElementById('timestamp');
59
60 // 最新データ監視
61 database.ref('rooms/latest').on('value', snapshot => {
62   container.innerHTML = ''; // 前の表示をクリア
63   const data = snapshot.val();
64   if (!data) return;
65
66   // タイムスタンプをまとめて表示
67   if (data.timestamp) {
68     const timestamp = new Date(data.timestamp);
69     tsDiv.textContent = "更新日時: " + timestamp.toLocaleString();
70   } else {
71     tsDiv.textContent = "";
72   }
73
74   for (let i = 1; i <= 18; i++) {
75     const roomKey = "room" + i;
76     const val = data[roomKey];
77
78     const div = document.createElement('div');
79     div.classList.add('room');
80
81     // 状態に応じて色と文字
82     let statusText;
83     if (val === 1) {
84       div.classList.add('bright');
85       statusText = "明るい";
86     } else if (val === 2) {
87       div.classList.add('dark');
88       statusText = "暗い";
89     } else {
90       div.classList.add('other');
```

```
91     statusText = val;
92   }
93
94   const roomNumber = document.createElement('div');
95   roomNumber.classList.add('room-number');
96   roomNumber.textContent = roomKey;
97
98   const roomStatus = document.createElement('div');
99   roomStatus.classList.add('room-status');
100  roomStatus.textContent = statusText;
101
102  div.appendChild(roomNumber);
103  div.appendChild(roomStatus);
104  container.appendChild(div);
105  }
106  });
107  </script>
108  </body>
109  </html>
```

webサイトの HTML2